White Paper

Intel Software Solutions Group

Igor Astakhov

Intel® AVX Realization of Infinite Impulse Response (IIR) Filter for Complex Float Data

Due to feedback dependency, it is very hard to implement SIMD optimization for Infinite Impulse Response (IIR) algorithm. This paper presents a new approach for IIR algorithm vectorization and shows a way that it can be implemented with Intel® Advanced Vector Extensions (AVX), an extension of the Intel® 64 Instruction Set Architecture. The presented approach is implemented with Intel AVX instructions and gives 1.6x speedup over Intel® SSE implementation that is 2x faster than the usual point-wise implementation.

April 2008

Intel Corporation

Contents

Figures

Introduction

This paper describes complex Infinite Impulse Response (IIR) filter implementation with Intel® AVX Single Instruction Multiple Data (SIMD) instruction set. The main difficulty of such realization is the feedback dependency that is the nature of IIR filter, that each next output point is dependent on the each previous output point. This paper describes the general approach for IIR vectorization and its specific realization based on the Intel AVX instruction set architecture (ISA). The performance aspects of such implementation are considered at the bottom.

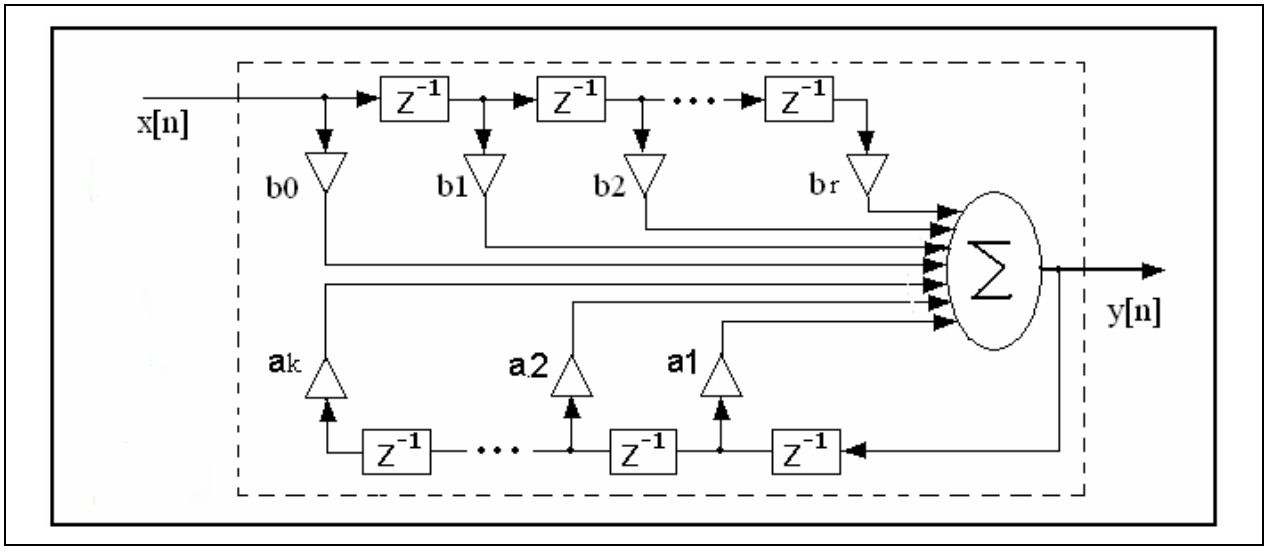
Intel® AVX Realization of IIR Filter for Complex Float Data

Let us take a look at the well-known IIR filter formula:

Equation 1

$$y_n = \sum_{i=0}^{order} b_i \cdot x_{n-i} - \sum_{j=1}^{order} a_j \cdot y_{n-j}$$

Figure 1. Structure of an Arbitrary Order IIR Filter



Where "*a*" and "*b*" are IIR coefficients, "*x*" – input data vector, "*y*" – output vector, "order" – filter order. There is a strong dependency of an output value on a set of previous output values. All further transformations have a goal to remove this feedback – the dependency of each output point on the previous points. We take into account the float data type and Intel AVX instruction width. Thus, we are interest in the unrolling of the computation loop with factor of 8 (or 4 if complex data type is processed). Due to universal approach, this method can be applied to any other data type – for example, real or double data with the certain unrolling factor. All further exposition is about complex float data type – therefore all additions, subtractions and multiplications below mean complex arithmetic operations. Therefore, our purpose is recalculation of the "*a*"

coefficients for unrolling on four the "*y*" calculations (to remove feedback latency). Equation 1) may be represented as:

Equation 2

$$y_n = F_x(n) - \sum_{j=1}^{order} a_j \cdot y_{n-j}$$

Where $F_x(n)$ is a function which depends only on "*x*" and it is simple Finite Impulse Response (FIR) filter. We shall do two-step successive filtering: at first by "*x*" (FIR), then by "*y*". FIR part for any 4 points can be represented with the corresponding formulas (assuming here for simplicity *"order"=k+1*):

Equation 3

$$F_x(n+0) = b_0 \cdot x_n + b_1 \cdot x_{n-1} + ... + b_k \cdot x_{n-k}$$

$$F_x(n+1) = b_0 \cdot x_{n+1} + b_1 \cdot x_n + ... + b_k \cdot x_{n-k+1}$$

$$F_x(n+2) = b_0 \cdot x_{n+2} + b_1 \cdot x_{n+1} + ... + b_k \cdot x_{n-k+2}$$

$$F_x(n+3) = b_0 \cdot x_{n+3} + b_1 \cdot x_{n+2} + ... + b_k \cdot x_{n-k+3}$$

The FIR part of IIR has relatively simple realization with using Intel AVX and unrolling on 8 (4 complex numbers) by vector length. Eight copies of each "*b.re*" & "*b.im*" tap aligned on 32-byte boundary are stored in IIR State structure, thus it is common for Intel AVX algorithm (see Initialization Function):

Figure 2. FIR–Part Coefficients Configuration

```
b0.re, b0.re, b0.re, b0.re, b0.re, b0.re, b0.re, b0.re
-b0.im, b0.im,-b0.im, b0.im,-b0.im, b0.im,-b0.im, b0.im
 b1.re, b1.re, b1.re, b1.re, b1.re, b1.re, b1.re, b1.re
-b1.im, b1.im,-b1.im, b1.im,-b1.im, b1.im,-b1.im, b1.im
……..
 bk.re, bk.re, bk.re, bk.re, bk.re, bk.re, bk.re, bk.re
-bk.im, bk.im,-bk.im, bk.im,-bk.im, bk.im,-bk.im, bk.im
```

Below is a block scheme for the FIR part of IIR algorithm implemented via new 256-bit registers (the full assembler code is presented in Assembler Function for B Coefficients):

Figure 3. FIR–Part of Algorithm

Load VMM reg = {$x_3.im$, $x_3.re$, $x_2.im$, $x_2.re$, $x_1.im$, $x_1.re$, $x_0.im$, $x_0.re$}
vmulps {$b_k.re$, $b_k.re$, $b_k.re$, $b_k.re$, $b_k.re$, $b_k.re$, $b_k.re$, $b_k.re$}
+
Vshufps VMM reg = {$x_3.re$, $x_3.im$, $x_2.re$, $x_2.im$, $x_1.re$, $x_1.im$, $x_0.re$, $x_0.im$}
vmulps {$-b_k.im$, $b_k.im$,$-b_k.im$, $b_k.im$,$-b_k.im$, $b_k.im$,$-b_k.im$, $b_k.im$}
+
……
……
+
Load VMM reg = {$x_{3+k}.im$, $x_{3+k}.re$, $x_{2+k}.im$, $x_{2+k}.re$, $x_{1+k}.im$, $x_{1+k}.re$, $x_k.im$, $x_k.re$}
vmulps {$b_0.re$, $b_0.re$, $b_0.re$, $b_0.re$, $b_0.re$, $b_0.re$, $b_0.re$, $b_0.re$}
+
Vshufps VMM reg = {$x_{3+k}.re$, $x_{3+k}.im$, $x_{2+k}.re$, $x_{2+k}.im$, $x_{1+k}.re$, $x_{1+k}.im$, $x_k.re$, $x_k.im$}
vmulps {$-b_0.im$, $b_0.im$,$-b_0.im$, $b_0.im$,$-b_0.im$, $b_0.im$,$-b_0.im$, $b_0.im$}
=
$F_x(n+0).re,im$, $F_x(n+1).re,im$, $F_x(n+2).re,im$, $F_x(n+3).re,im$

Temporally store to destination vector.

After the first step all $F_x(n)$ should be saved at destination vector and we will have:

Equation 4

$$y_n = -a_1 \cdot y_{n-1} - a_2 \cdot y_{n-2} - \ldots - a_k \cdot y_{n-k} + F_x(n+0)$$

$$y_{n+1} = -a_1 \cdot y_n - a_2 \cdot y_{n-1} - \ldots - a_k \cdot y_{n-k+1} + F_x(n+1)$$

$$y_{n+2} = -a_1 \cdot y_{n+1} - a_2 \cdot y_n - \ldots - a_k \cdot y_{n-k+2} + F_x(n+2)$$

$$y_{n+3} = -a_1 \cdot y_{n+2} - a_2 \cdot y_{n+1} - \ldots - a_k \cdot y_{n-k+3} + F_x(n+3)$$

To remove feedback latency, let us substitute the y_n from the first line to the next three lines, then y_{n+1} from the second line to the next two lines, y_{n+2} from the third - to the line number four. For example point y_{n+2}:

Equation 5

$$\begin{aligned}
&y_{n+2} = -a_1 \cdot y_{n+1} - a_2 \cdot y_n - \ldots - a_k \cdot y_{n-k+2} + F_x(n+2) = \\
&- a_1 \cdot (-a_1 \cdot (-a_1 \cdot y_{n-1} - a_2 \cdot y_{n-2} - \ldots - a_k \cdot y_{n-k} + F_x(n+0)) - a_2 \cdot y_{n-1} - \ldots - \\
&- a_k \cdot y_{n-k+1} + F_x(n+1)) - a_2 \cdot (-a_1 \cdot y_{n-1} - a_2 \cdot y_{n-2} - \ldots - \\
&- a_k \cdot y_{n-k} + F_x(n+0)) - \ldots - a_k \cdot y_{n-k+2} + F_x(n+2) = \\
&= (2a_2a_1 - a_1^3) \cdot y_{n-1} + (a_2^2 - a_1^2 a_2) \cdot y_{n-2} + \ldots + (a_1^2 - a_2) \cdot F_x(n+0) - a_1 \cdot F_x(n+1) + F_x(n+2)
\end{aligned}$$

At the left of the first four lines we'll have new output points – y_n , y_{n+1} , y_{n+2} , y_{n+3}, and at the right - the functions from the "old" points (y_{n-1} …y_{n-k}) and from $F_x(n+0)…F_x(n+3)$ with some new coefficients, which are calculated during pState structure initializing (see Initialization Function):

Equation 6

$$\begin{vmatrix} a_{01} \\ a_{02} \\ \dots \\ a_{0k} \end{vmatrix} = \begin{vmatrix} a_1 \\ a_2 \\ \dots \\ a_k \end{vmatrix}$$

Equation 7

$$\begin{vmatrix} a_{11} \\ a_{12} \\ \dots \\ a_{1k} \end{vmatrix} = a_1 \begin{vmatrix} a_{01} \\ a_{02} \\ \dots \\ a_{0k} \end{vmatrix} + \begin{vmatrix} 0 \\ a_1 \\ \dots \\ a_{k-1} \end{vmatrix}$$

Equation 8

$$\begin{vmatrix} a_{21} \\ a_{22} \\ \dots \\ a_{2k} \end{vmatrix} = a_1 \begin{vmatrix} a_{11} \\ a_{12} \\ \dots \\ a_{1k} \end{vmatrix} + a_2 \begin{vmatrix} a_{01} \\ a_{02} \\ \dots \\ a_{0k} \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ \dots \\ a_{k-2} \end{vmatrix}$$

Equation 9

$$\begin{vmatrix} a_{31} \\ a_{32} \\ \dots \\ a_{3k} \end{vmatrix} = a_1 \begin{vmatrix} a_{21} \\ a_{22} \\ \dots \\ a_{2k} \end{vmatrix} + a_2 \begin{vmatrix} a_{11} \\ a_{12} \\ \dots \\ a_{1k} \end{vmatrix} + a_3 \begin{vmatrix} a_{01} \\ a_{02} \\ \dots \\ a_{0k} \end{vmatrix} + \begin{vmatrix} 0 \\ 0 \\ \dots \\ a_{k-3} \end{vmatrix}$$

For complex float data type and GSSE operations the unrolling factor should be 4. Now we have 4 new sets of complex "*a*" coefficients (of "*y*") for a bundle of size four:

- a_{0n} for the first point of a bundle – Equation 6;
- a_{1n} for the second point of a bundle – Equation 7;
- a_{2n} for the third point of a bundle – Equation 8;
- a_{3n} for the fourth point of a bundle – Equation 9.

And also we have to do some extra calculations for $F_x(n+0)._{re,im}$, $F_x(n+1)._{re,im}$, $F_x(n+2)._{re,im}$, $F_x(n+3)._{re,im}$ – coefficients for the each point of a bundle are the next:

Figure 4. The Extra Triangle of Coefficients for IIR-Part

1.	*0,*	*0,*	*0,*	*1;*
2.	a_{01},	*0,*	*0,*	*1;*
3.	a_{11},	a_{01},	*0,*	*1;*
4.	a_{21},	a_{11},	a_{01},	*1;*

In the same manner as for the first (FIR) step all new coefficients are calculated at the init stage (see Initialization Function) and are stored in the natural for Intel AVX order with the appropriate alignment (the part with all "1" is not required and is missed):

Figure 5. IIR–Part Coefficients Configuration

$a_{01.re,im}$, $a_{11.re,im}$, $a_{21.re,im}$, $a_{31.re,im}$

$a_{02.re,im}$, $a_{12.re,im}$, $a_{22.re,im}$, $a_{32.re,im}$

……

$a_{0k.re,im}$, $a_{1k.re,im}$, $a_{2k.re,im}$, $a_{3k.re,im}$

$0_{.re,im}$, $a_{01.re,im}$, $a_{11.re,im}$, $a_{21.re,im}$

$0_{.re,im}$, $0_{.re,im}$, $a_{01.re,im}$, $a_{11.re,im}$

$0_{.re,im}$, $0_{.re,im}$, $0_{.re,im}$, $a_{01.re,im}$

Therefore for each 4-output points we have 10 extra complex additions and 6 extra complex multiplications that are the algorithm cost for the feedback dependency removing:

Figure 6. IIR–Part of Algorithm

Vbroadcast to VMM = $\{y_{n-1.re}, y_{n-1.re}, y_{n-1.re}, y_{n-1.re}, y_{n-1.re}, y_{n-1.re}, y_{n-1.re}, y_{n-1.re}\}$
vmulps $\{a_{01.re}, a_{01.im}, a_{11.re}, a_{11.im}, a_{21.re}, a_{21.im}, a_{31.re}, a_{31.im}\}$ +

Vbroadcast to VMM = $\{y_{n-1.im}, y_{n-1.im}, y_{n-1.im}, y_{n-1.im}, y_{n-1.im}, y_{n-1.im}, y_{n-1.im}, y_{n-1.im}\}$
vmulps $\{a_{01.im}, a_{01.re}, a_{11.im}, a_{11.re}, a_{21.im}, a_{21.re}, a_{31.im}, a_{31.re}\}$ +

…… …… +

Vbroadcast to VMM = $\{y_{n-k.re}, y_{n-k.re}, y_{n-k.re}, y_{n-k.re}, y_{n-k.re}, y_{n-k.re}, y_{n-k.re}, y_{n-k.re}\}$
vmulps $\{a_{0k.re}, a_{0k.im}, a_{1k.re}, a_{1k.im}, a_{2k.re}, a_{2k.im}, a_{3k.re}, a_{3k.im}\}$ +

Vbroadcast to VMM = $\{y_{n-k.im}, y_{n-k.im}, y_{n-k.im}, y_{n-k.im}, y_{n-k.im}, y_{n-k.im}, y_{n-k.im}, y_{n-k.im}\}$
vmulps $\{a_{0k.im}, a_{0k.re}, a_{1k.im}, a_{1k.re}, a_{2k.im}, a_{2k.re}, a_{3k.im}, a_{3k.re}\}$ +

Vbroadcast to VMM = $\{F_x(n+0), F_x(n+0), F_x(n+0), F_x(n+0)\}$re.im
vmulps $\{a_{0k.re,im}, a_{1k.re,im}, a_{2k.re,im}, a_{3k.re,im}\}$ + *(extra)*

Vbroadcast to VMM = $\{F_x(n+1), F_x(n+1), F_x(n+1), F_x(n+1)\}$re.im
vmulps $\{0_{.re,im}, a_{01.re,im}, a_{11.re,im}, a_{21.re,im}\}$ + *(extra)*

Vbroadcast to VMM = $\{F_x(n+2), F_x(n+2), F_x(n+2), F_x(n+2)\}$re.im
vmulps $\{0_{.re,im}, 0_{.re,im}, a_{01.re,im}, a_{11.re,im}\}$ + *(extra)*

Vbroadcast to VMM = $\{F_x(n+3), F_x(n+3), F_x(n+3), F_x(n+3)\}$re.im
vmulps $\{0_{.re,im}, 0_{.re,im}, 0_{.re,im}, a_{01.re,im}\}$ + *(extra)*

Vload to VMM = $\{F_x(n+3), F_x(n+2), F_x(n+1), F_x(n+0)\}$re.im =

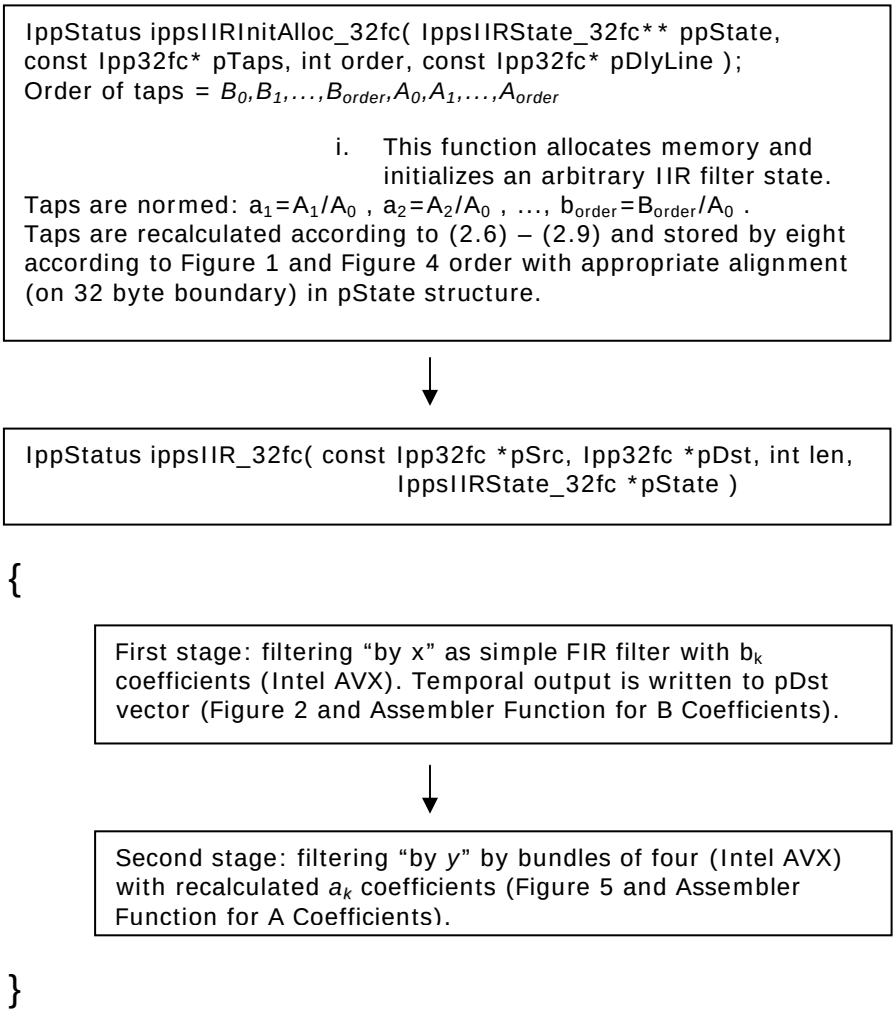
Vstore VMM = $\{Yn+0, Yn+1, Yn+2, Yn+3\}re.im$

Temporally store to destination vector

This approach allows to remove feedback for the whole bundle and to improve IIR filter performance on all SIMD architectures more than two times.

Below is the full IIR block scheme for the algorithm described above that is used in Intel® Integrated Performance Primitives (Intel® IPP) libraries.

Figure 7. Full IIR Algorithm Block–Scheme



Conclusion

Because of feedback dependency the previous realizations of IIR filter function for IA32 architecture were point-wise, not vector. It means that vector function performance per output point is almost the same as for one-point function. The unrolling was done “by filter order”, not by vector length. The visible performance improvement in comparison with “C” code was only for filter orders 32 and higher. Such large orders of the IIR filters are not practically used in signal processing. For the orders less than 32 the usual implementation of IIR filter with Intel SSE does not provide any performance gain because of the latency of the instructions, which is accumulated in the common scheme with feedback dependency. The presented approach (the feedback dependency removing) implemented with Intel AVX instructions gives ~1.6x speedup for the filter of order 6 over the analogous Intel SSE code (it must be taken into account that this ratio is obtained under architectural software simulator and actual hardware numbers may differ).

Appendix A: Function Code

Initialization Function

```
extern IppStatus ownsIIRSetTaps_32fc( const Ipp32fc* pTaps,
                                                    IppsIIRState_32fc* pState )
{
    int i, order = pState->order;
    Ipp64fc a0inv;
    Ipp32f *pA, *pB, *pO;
    Ipp32fc kA, kB, kC, *pTmp;
    int j;

    if(( 0 == pTaps[order+1].re )&&( 0 == pTaps[order+1].im )){
        return ippStsDivByZeroErr;
    }
    a0inv.re = SQR( (Ipp64f)pTaps[order+1].re ) +
               SQR( (Ipp64f)pTaps[order+1].im );
    a0inv.im = -pTaps[order+1].im / a0inv.re;
    a0inv.re =  pTaps[order+1].re / a0inv.re;
    pState->taps[0].re = (Ipp32f)( MUL_RE( pTaps[0], a0inv ));
    pState->taps[0].im = (Ipp32f)( MUL_IM( pTaps[0], a0inv ));
    for( i = 1; i <= order; i++ ){
        pState->taps[i].re = (Ipp32f)( MUL_RE( pTaps[i], a0inv ));
        pState->taps[i].im = (Ipp32f)( MUL_IM( pTaps[i], a0inv ));
        pState->taps[order+i].re = (Ipp32f)( MUL_RE( pTaps[order+i+1], a0inv ));
        pState->taps[order+i].im = (Ipp32f)( MUL_IM( pTaps[order+i+1], a0inv ));
    }
/* ************ SetTaps section for the optimized GSSE code ************ */
    pB = pState->btaps;
    pA = pState->ataps;
    pO = pState->otaps;
    for( i = 0; i <= order; i++ ){
        j = 16 * i;
/* taps for "FIR" part (by "x"): Tx.re, Tx.re, Tx.re, Tx.re, Tx.re, Tx.re, Tx.re, Tx.re,
   unrolling on 8 for YMM:      -Tx.im, Tx.im,-Tx.im, Tx.im,-Tx.im, Tx.im,-Tx.im, Tx.im, */
        pB[j+0] = pB[j+1] = pB[j+2] = pB[j+3] = pB[j+4] = pB[j+5] = pB[j+6] = pB[j+7] =
                                      (Ipp32f)( MUL_RE( a0inv, pTaps[i] ));
        pB[j+9] = pB[j+11] = pB[j+13] = pB[j+15] =(Ipp32f)( MUL_IM( a0inv, pTaps[i] ));
        pB[j+8] = pB[j+10] = pB[j+12] = pB[j+14] = -pB[j+9];
    }
    if( order > 0 ){
        for( i = 0; i < order; i++ ){
            j = 16 * i;
            pA[j+0] = -(Ipp32f)( MUL_RE( a0inv, pTaps[order+2+i] ));    /* A1[0].re */
            pA[j+1] = -(Ipp32f)( MUL_IM( a0inv, pTaps[order+2+i] ));    /* A1[0].im */
            pA[j+2] =  pA[0] * pA[j+0] - pA[1] * pA[j+1];               /* A1[1].re */
            pA[j+3] =  pA[0] * pA[j+1] + pA[1] * pA[j+0];               /* A1[1].im */
            pA[j+8] = -pA[j+1];                                         /*-A1[0].im */
            pA[j+9] =  pA[j+0];                                         /* A1[0].re */
            pA[j+10]= -pA[j+3];                                         /*-A1[1].im */
            pA[j+11]=  pA[j+2];                                         /* A1[1].re */
        }
        kA.re = pA[0];
        kA.im = pA[1];
        if( order > 1 ){
            kB.re = pA[16];                                             /* A2[0].re */
            kB.im = pA[17];                                             /* A2[0].im */
        } else {
            kB.re = kB.im = 0.f;
        }
        if( order > 2 ){
            kC.re = pA[32];                                             /* A3[0].re */
            kC.im = pA[33];                                             /* A3[0].im */
        } else {
```

```
            kC.re = kC.im = 0.f;
        }
        for( i = 0; i < order - 1; i++ ){
            j = 16 * i;
            pA[j+2] += pA[j+16];                                        /* A1[1].re */
            pA[j+3] += pA[j+17];                                        /* A1[1].im */
            pA[j+10] = -pA[j+3];                                        /*-A1[1].im */
            pA[j+11] = pA[j+2];                                         /* A1[1].re */
        }
        for( i = 0; i < order; i++ ){
            j = 16 * i;
            pA[j+4] = kA.re * pA[j+2] - kA.im * pA[j+3] +
                      kB.re * pA[j]   - kB.im * pA[j+1];                /* A1[2].re */
            pA[j+5] = kA.re * pA[j+3] + kA.im * pA[j+2] +
                      kB.re * pA[j+1] + kB.im * pA[j];                  /* A1[2].im */
            pA[j+12] = -pA[j+5];                                        /*-A1[2].im */
            pA[j+13] =  pA[j+4];                                        /* A1[2].re */
        }
        for( i = 0; i < order - 2; i++ ){
            j = 16 * i;
            pA[j+4]  += pA[j+32];                                       /* A1[2].re */
            pA[j+5]  += pA[j+33];                                       /* A1[2].im */
            pA[j+12] = -pA[j+5];                                        /*-A1[2].im */
            pA[j+13] =  pA[j+4];                                        /* A1[2].re */
        }
        for( i = 0; i < order; i++ ){
            j = 16 * i;
            pA[j+6]  = kA.re * pA[j+4] - kA.im * pA[j+5] +
                       kB.re * pA[j+2] - kB.im * pA[j+3] +
                       kC.re * pA[j]   - kC.im * pA[j+1];               /* A1[3].re */
            pA[j+7]  = kA.re * pA[j+5] + kA.im * pA[j+4] +
                       kB.re * pA[j+3] + kB.im * pA[j+2] +
                       kC.re * pA[j+1] + kC.im * pA[j];                 /* A1[3].im */
            pA[j+14] =- pA[j+7];                                        /*-A1[3].im */
            pA[j+15]  = pA[j+6];                                        /* A1[3].re */
        }
        for( i = 0; i < order - 3; i++ ){
            j = 16 * i;
            pA[j+6]  += pA[j+48];                                       /* A1[3].re */
            pA[j+7]  += pA[j+49];                                       /* A1[3].im */
            pA[j+14] =- pA[j+7];                                        /*-A1[3].im */
            pA[j+15]  = pA[j+6];                                        /* A1[3].re */
        }
/*      For the "Fx" coefficients (for the vector {Fx[0], Fx[1], Fx[2], Fx[3]}):
//      line 1:     0,     0,     0,     1;
//      line 2: A1[0],     0,     0,     1;
//      line 3: A1[1], A1[0],     0,     1;
//      line 4: A1[2], A1[1], A1[0],     1; */
        pA[16*order+0] = 0.f;               /* 0.re     */
        pA[16*order+1] = 0.f;               /* 0.im     */
        pA[16*order+2] = pA[0];             /* A1[0].re */
        pA[16*order+3] = pA[1];             /* A1[0].im */
        pA[16*order+4] = pA[2];             /* A1[1].re */
        pA[16*order+5] = pA[3];             /* A1[1].im */
        pA[16*order+6] = pA[4];             /* A1[2].re */
        pA[16*order+7] = pA[5];             /* A1[2].im */

        pA[16*order+8]  = 0.f;              /* 0.im     */
        pA[16*order+9]  = 0.f;              /* 0.re     */
        pA[16*order+10] = pA[8];            /*-A1[0].im */
        pA[16*order+11] = pA[9];            /* A1[0].re */
        pA[16*order+12] = pA[10];           /*-A1[1].im */
        pA[16*order+13] = pA[11];           /* A1[1].re */
        pA[16*order+14] = pA[12];           /*-A1[2].im */
        pA[16*order+15] = pA[13];           /* A1[2].re */

        pA[16*order+16] = 0.f;              /* 0.re     */
        pA[16*order+17] = 0.f;              /* 0.im     */
        pA[16*order+18] = 0.f;              /* 0.re     */
        pA[16*order+19] = 0.f;              /* 0.im     */
        pA[16*order+20] = pA[0];            /* A1[0].re */
```

```
        pA[16*order+21] = pA[1];            /* A1[0].im */
        pA[16*order+22] = pA[2];            /* A1[1].re */
        pA[16*order+23] = pA[3];            /* A1[1].im */

        pA[16*order+24] = 0.f;              /* 0.im     */
        pA[16*order+25] = 0.f;              /* 0.re     */
        pA[16*order+26] = 0.f;              /* 0.im     */
        pA[16*order+27] = 0.f;              /* 0.re     */
        pA[16*order+28] = pA[8];            /*-A1[0].im */
        pA[16*order+29] = pA[9];            /* A1[0].re */
        pA[16*order+30] = pA[10];           /*-A1[1].im */
        pA[16*order+31] = pA[11];           /* A1[1].re */

        pA[16*order+32] = 0.f;              /* 0.re     */
        pA[16*order+33] = 0.f;              /* 0.im     */
        pA[16*order+34] = 0.f;              /* 0.re     */
        pA[16*order+35] = 0.f;              /* 0.im     */
        pA[16*order+36] = 0.f;              /* 0.re     */
        pA[16*order+37] = 0.f;              /* 0.im     */
        pA[16*order+38] = pA[0];            /* A1[0].re */
        pA[16*order+39] = pA[1];            /* A1[0].im */

        pA[16*order+40] = 0.f;              /* 0.im     */
        pA[16*order+41] = 0.f;              /* 0.re     */
        pA[16*order+42] = 0.f;              /* 0.im     */
        pA[16*order+43] = 0.f;              /* 0.re     */
        pA[16*order+44] = 0.f;              /* 0.im     */
        pA[16*order+45] = 0.f;              /* 0.re     */
        pA[16*order+46] = pA[8];            /*-A1[0].im */
        pA[16*order+47] = pA[9];            /* A1[0].re */
    }
    pTmp = pState->taps;
    /* taps for "One" (point-wize) function in order suitable for SSE operations */
    pO[0] = pO[1] = pTmp[0].re; pO[2] = -pTmp[0].im; pO[3] = pTmp[0].im;
    for( i = 1; i <= ( order & (~1)); i += 2 ){
        j = 8 * ( i - 1 );
        pO[j+4]  =  pO[j+5] = pTmp[i].re;
        pO[j+6]  =  pO[j+7] = pTmp[i+1].re;
        pO[j+8]  = -pTmp[i].im;
        pO[j+9]  =  pTmp[i].im;
        pO[j+10] = -pTmp[i+1].im;
        pO[j+11] =  pTmp[i+1].im;
        pO[j+12] =  pO[j+17] = -pTmp[order+i].re;
        pO[j+14] =  pO[j+19] = -pTmp[order+i+1].re;
        pO[j+13] = -pTmp[order+i].im;
        pO[j+16] =  pTmp[order+i].im;
        pO[j+15] = -pTmp[order+i+1].im;
        pO[j+18] =  pTmp[order+i+1].im;
    }
    if( order & 1 ){
        j = 8 * ( order - 1 );
        pO[j+4]  =  pO[j+5] = pTmp[order].re;
        pO[j+8]  = -pTmp[order].im;
        pO[j+9]  =  pTmp[order].im;
        pO[j+12] =  pO[j+17] = -pTmp[2*order].re;
        pO[j+13] = -pTmp[2*order].im;
        pO[j+16] =  pTmp[2*order].im;
    }
/* *************************** end of section **************************** */
    return ippStsNoErr;
}
```

Filtering Function

```
static IppStatus ownsIIRAR_32fc( Ipp32fc *pSrc, Ipp32fc *pDst, int len,
                                                    IppsIIRState_32fc *pState )
{
    int      i, n;
    Ipp32fc *pDly  = pState->dlyLine;
    Ipp32f  *pA    = pState->ataps;
```

```
    Ipp32f  *pB    = pState->btaps;
    Ipp32fc *pBuf  = pState->pBuf;
    Ipp32fc *pTaps = pState->taps;
    int      order = pState->order;

    if( len <= 4 * order ){
        for( i = 0; i < len; i++ ){
            ippsIIROne_32fc( pSrc[i], &pDst[i], pState );
        }
    } else {
        /* two-steps successive filtering: by x, then by y...*/
        /* 1-st stage: only "B" coeff. (by "x") at first */
        owngIIRxAR_32fc( pSrc, pBuf, ( len - order ), pB, order );
        /* 2-nd stage: first "order" points must be calculated from
                       the current delay line */
        for( i = 0; i < order; i++ ){
            ippsIIROne_32fc( pSrc[i], &pDst[i], pState );
        }
        for( i = 0; i < order; i++ ){
            pDly[i].re = pDly[i].im = 0.f;
            for( n = order - i; n > 0; n-- ){
                pDly[i].re += MUL_RE( pTaps[n+i], pSrc[len-n] );
                pDly[i].im += MUL_IM( pTaps[n+i], pSrc[len-n] );
            }
        }
        /* 3-rd stage: then "A" coeff. (by "y") */
        owngIIRyAR_32fc( pBuf, pDst, ( len - order ), pA, order );
        /* last stage: form new delay line */
        for( i = 0; i < order; i++ ){
            for( n = order - i; n > 0; n-- ){
                pDly[i].re -= MUL_RE( pTaps[order+n+i], pDst[len-n] );
                pDly[i].im -= MUL_IM( pTaps[order+n+i], pDst[len-n] );
            }
        }
    }
    return ippStsNoErr;
}
```

Assembler Function for B Coefficients

```
;#########################################################################
;##   ownsIIRxAR_32fc( Ipp32fc *pSrc, Ipp32fc *pDst, int len, Ipp32f *pTaps  ##
;##                                    int order );                       ##
;#########################################################################
;   Declare input variables

pSrc        equ rdi
pDst        equ rsi
len         equ rdx
pTaps       equ rcx
order       equ r8

ALIGN 16

; Lib = E9
; Caller = ippsIIR_32fc
; Caller = ippsIIR_32fc_I

ownsIIRxAR_32fc PROC PUBLIC FRAME
        USES_GPR rbx, rsi, rdi,r10,r12,r13, r15
        LOCAL_FRAME = 0
        USES_XMM xmm6, xmm7, xmm8
        COMP_ABI 5
        movsxd      r8,r8d
        mov         rbx,pTaps
```

```
        mov         r11,pSrc
        mov         r10,pDst
        mov         r13,len
        mov         r15, len
        mov         rax,order
        cmp         rax,0
        je          Order0
        cmp         rax,1
        je          Order1
        cmp         rax,2
        je          Order2
        cmp         rax,3
        je          Order3
        jmp         OrderAny
Order0:
        and         r13,-8
        je          try1tail
        lea         r13,[r11+8*r13]
        test        r11,0Fh
        jne         doU01by8
doA01by8:
        vmovaps     xmm0,oword ptr [r11]
        vmovaps     xmm1,oword ptr [r11+10h]
        vmovaps     xmm2,oword ptr [r11+20h]
        vmovaps     xmm3,oword ptr [r11+30h]
        add         r11,40h
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm1,xmm1,oword ptr [rbx]
        vmulps      xmm2,xmm2,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm4,xmm4,oword ptr [rbx+20h]
        vmulps      xmm5,xmm5,oword ptr [rbx+20h]
        vmulps      xmm6,xmm6,oword ptr [rbx+20h]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm4
        vaddps      xmm1,xmm1,xmm5
        vaddps      xmm2,xmm2,xmm6
        vaddps      xmm3,xmm3,xmm7
        vmovaps     oword ptr [r10],xmm0
        vmovaps     oword ptr [r10+10h],xmm1
        vmovaps     oword ptr [r10+20h],xmm2
        vmovaps     oword ptr [r10+30h],xmm3
        add         r10,40h
        cmp         r11,r13
        jl          doA01by8
try1tail:
        mov         r13,len
        and         r13,7
        je          exit1
do1tail:
        vmovsd      xmm0,qword ptr [r11]
        add         r11,8
        vpshufd     xmm1,xmm0,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm1,xmm1,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm1
        vmovsd      qword ptr [r10],xmm0
        add         r10,8
        sub         r13,1
        ja          do1tail
exit1:
        REST_XMM
        REST_GPR
        ret
doU01by8:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm4,qword ptr [r11+8]
```

```
        vmovsd      xmm1,qword ptr [r11+10h]
        vmovsd      xmm5,qword ptr [r11+18h]
        vmovsd      xmm2,qword ptr [r11+20h]
        vmovsd      xmm6,qword ptr [r11+28h]
        vmovsd      xmm3,qword ptr [r11+30h]
        vmovsd      xmm7,qword ptr [r11+38h]
        add         r11,40h
        vmovlhps    xmm0,xmm0,xmm4
        vmovlhps    xmm1,xmm1,xmm5
        vmovlhps    xmm2,xmm2,xmm6
        vmovlhps    xmm3,xmm3,xmm7
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm1,xmm1,oword ptr [rbx]
        vmulps      xmm2,xmm2,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm4,xmm4,oword ptr [rbx+20h]
        vmulps      xmm5,xmm5,oword ptr [rbx+20h]
        vmulps      xmm6,xmm6,oword ptr [rbx+20h]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm4
        vaddps      xmm1,xmm1,xmm5
        vaddps      xmm2,xmm2,xmm6
        vaddps      xmm3,xmm3,xmm7
        vmovaps     oword ptr [r10],xmm0
        vmovaps     oword ptr [r10+10h],xmm1
        vmovaps     oword ptr [r10+20h],xmm2
        vmovaps     oword ptr [r10+30h],xmm3
        add         r10,40h
        cmp         r11,r13
        jl          doU01by8
        jmp         try1tail
Order1:
        and         r13,-16
        je          try2tail
        lea         r13,[r11+8*r13]
        test        r11,1Fh
        jne         doU02by8
        vmovaps     ymm12, oword ptr [rbx+40h]  ; t0.re, t0.re, t0.re, t0.re, t0.re, t0.re,
t0.re, t0.re
        vmovaps     ymm13, oword ptr [rbx+60h]  ; t0.im,-t0.im, t0.im,-t0.im, t0.im,-t0.im,
t0.im,-t0.im
        vmovaps     ymm14, oword ptr [rbx]      ; t1.re, t1.re, t1.re, t1.re, t1.re, t1.re,
t1.re, t1.re
        vmovaps     ymm15, oword ptr [rbx+20h]  ; t1.im,-t1.im, t1.im,-t1.im, t1.im,-t1.im,
t1.im,-t1.im
doA02by16:
        vmovaps     ymm0, oword ptr [r11]       ; x3.im, x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vmovaps     ymm4, oword ptr [r11+20h]   ; x7.im, x7.re, x6.im, x6.re, x5.im, x5.re,
x4.im, x4.re
        vmovsd      xmm5, qword ptr [r11+40h]   ;
x8.im, x8.re
        add         r11, 40h
        vperm2f128  ymm2, ymm0, ymm4, 21h       ; x5.im, x5.re, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm0, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vperm2f128  ymm5, ymm4, ymm5, 21h       ; xxxxx, xxxxx, x8.im, x8.re, x7.im, x7.re,
x6.im, x6.re
        vblendps    ymm5, ymm5, ymm4, 0CCh      ; x7.im, x7.re, x8.im, x8.re, x5.im, x5.re,
x6.im, x6.re
        vshufps     ymm5, ymm5, ymm5, 4Eh       ; x8.im, x8.re, x7.im, x7.re, x6.im, x6.re,
x5.im, x5.re
        vshufps     ymm6, ymm0, ymm0, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
```

```
        vshufps     ymm7, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vshufps     ymm10, ymm4, ymm4, 0B1h     ; x7.re, x7.im, x6.re, x6.im, x5.re, x5.im,
x4.re, x4.im
        vshufps     ymm11, ymm5, ymm5, 0B1h     ; x8.re, x8.im, x7.re, x7.im, x6.re, x6.im,
x5.re, x5.im
        vmulps      ymm0, ymm0, ymm12           ; x3-x0.re * t0.re
        vmulps      ymm6, ymm6, ymm13           ; x3-x0.im * t0.im
        vaddps      ymm0, ymm0, ymm6
        vmulps      ymm1, ymm1, ymm14           ; x4-x1.re * t1.re
        vmulps      ymm7, ymm7, ymm15           ; x4-x1.im * t1.im
        vaddps      ymm1, ymm7, ymm1
        vaddps      ymm0, ymm0, ymm1            ; Y3.im, Y3.re, Y2.im, Y2.re, Y1.im, Y1.re,
Y0.im, Y0.re
        vmovaps     oword ptr [r10], ymm0
        vmulps      ymm4, ymm4, ymm12           ; x7-x4.re * t0.re
        vmulps      ymm10, ymm10, ymm13         ; x7-x4.im * t0.im
        vaddps      ymm4, ymm10, ymm4
        vmulps      ymm5, ymm5, ymm14           ; x8-x5.re * t1.re
        vmulps      ymm11, ymm11, ymm15         ; x8-x5.im * t1.im
        vaddps      ymm5, ymm11, ymm5
        vaddps      ymm4, ymm5, ymm4            ; Y7.im, Y7.re, Y6.im, Y6.re, Y5.im, Y5.re,
Y4.im, Y4.re
        vmovaps     oword ptr [r10+20h], ymm4
        add         r10, 40h
        cmp         r11, r13
        jl          doA02by16                   ; 33 instructions loop ~4 instr/out point
try2tail:
        mov         r13,len
        and         r13,15
        je          exit2
do2tail:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        add         r11,8
        vpshufd     xmm2,xmm0,0B1h
        vpshufd     xmm3,xmm1,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+40h]
        vmulps      xmm2,xmm2,oword ptr [rbx+60h]
        vmulps      xmm1,xmm1,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm2
        vaddps      xmm1,xmm1,xmm3
        vaddps      xmm0,xmm0,xmm1
        vmovsd      qword ptr [r10],xmm0
        add         r10,8
        sub         r13,1
        ja          do2tail
exit2:
        REST_XMM
        REST_GPR
        ret
doU02by8:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovsd      xmm3,qword ptr [r11+18h]
        vmovsd      xmm4,qword ptr [r11+20h]
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vmovlhps    xmm3,xmm3,xmm4
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+40h]
        vmulps      xmm2,xmm2,oword ptr [rbx+40h]
        vmulps      xmm4,xmm4,oword ptr [rbx+60h]
        vmulps      xmm6,xmm6,oword ptr [rbx+60h]
        vmulps      xmm1,xmm1,oword ptr [rbx]
```

```
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm5,xmm5,oword ptr [rbx+20h]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm4
        vaddps      xmm2,xmm2,xmm6
        vaddps      xmm1,xmm1,xmm5
        vaddps      xmm3,xmm3,xmm7
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vmovaps     oword ptr [r10],xmm0
        vmovaps     oword ptr [r10+10h],xmm2
        vmovsd      xmm0,qword ptr [r11+20h]
        vmovsd      xmm1,qword ptr [r11+28h]
        vmovsd      xmm2,qword ptr [r11+30h]
        vmovsd      xmm3,qword ptr [r11+38h]
        vmovsd      xmm4,qword ptr [r11+40h]
        add         r11,40h
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vmovlhps    xmm3,xmm3,xmm4
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+40h]
        vmulps      xmm2,xmm2,oword ptr [rbx+40h]
        vmulps      xmm4,xmm4,oword ptr [rbx+60h]
        vmulps      xmm6,xmm6,oword ptr [rbx+60h]
        vmulps      xmm1,xmm1,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm5,xmm5,oword ptr [rbx+20h]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm4
        vaddps      xmm2,xmm2,xmm6
        vaddps      xmm1,xmm1,xmm5
        vaddps      xmm3,xmm3,xmm7
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vmovaps     oword ptr [r10+20h],xmm0
        vmovaps     oword ptr [r10+30h],xmm2
        add         r10,40h
        cmp         r11,r13
        jl          doU02by8
        jmp         try2tail
Order2:
        and         r13,-8
        je          try3tail
        lea         r13,[r11+8*r13]
        test        r11,1Fh
        jne         doU03by4
        vmovaps     ymm12, oword ptr [rbx+80h]  ; t0.re, t0.re, t0.re, t0.re, t0.re, t0.re,
t0.re, t0.re
        vmovaps     ymm13, oword ptr [rbx+0A0h] ; t0.im,-t0.im, t0.im,-t0.im, t0.im,-t0.im,
t0.im,-t0.im
        vmovaps     ymm14, oword ptr [rbx+40h]  ; t1.re, t1.re, t1.re, t1.re, t1.re, t1.re,
t1.re, t1.re
        vmovaps     ymm15, oword ptr [rbx+60h]  ; t1.im,-t1.im, t1.im,-t1.im, t1.im,-t1.im,
t1.im,-t1.im
doA03by8:
        vmovaps     ymm0, oword ptr [r11]       ; x3.im, x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vmovaps     ymm3, oword ptr [r11+20h]   ; x7.im, x7.re, x6.im, x6.re, x5.im, x5.re,
x4.im, x4.re
        vmovaps     xmm4, oword ptr [r11+40h]   ;                             x9.im, x9.re,
x8.im, x8.re
        add         r11, 40h
        vperm2f128  ymm2, ymm0, ymm3, 21h       ; x5.im, x5.re, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm0, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
```

```
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vperm2f128  ymm5, ymm3, ymm4, 21h       ; x9.im, x9.re, x8.im, x8.re, x7.im, x7.re,
x6.im, x6.re
        vblendps    ymm4, ymm5, ymm3, 0CCh      ; x7.im, x7.re, x8.im, x8.re, x5.im, x5.re,
x6.im, x6.re
        vshufps     ymm4, ymm4, ymm4, 4Eh       ; x8.im, x8.re, x7.im, x7.re, x6.im, x6.re,
x5.im, x5.re
        vshufps     ymm6, ymm0, ymm0, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vshufps     ymm7, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vshufps     ymm8, ymm2, ymm2, 0B1h      ; x5.re, x5.im, x4.re, x4.im, x3.re, x3.im,
x2.re, x2.im
        vshufps     ymm9, ymm3, ymm3, 0B1h      ; x7.re, x7.im, x6.re, x6.im, x5.re, x5.im,
x4.re, x4.im
        vshufps     ymm10, ymm4, ymm4, 0B1h     ; x8.re, x8.im, x7.re, x7.im, x6.re, x6.im,
x5.re, x5.im
        vshufps     ymm11, ymm5, ymm5, 0B1h     ; x9.re, x9.im, x8.re, x8.im, x7.re, x7.im,
x6.re, x6.im
        vmulps      ymm0, ymm0, ymm12           ; x3-x0.re * t0.re
        vmulps      ymm6, ymm6, ymm13           ; x3-x0.im * t0.im
        vmulps      ymm3, ymm3, ymm12           ; x7-x4.re * t0.re
        vmulps      ymm9, ymm9, ymm13           ; x7-x4.im * t0.im
        vmulps      ymm1, ymm1, ymm14           ; x4-x1.re * t1.re
        vmulps      ymm7, ymm7, ymm15           ; x4-x1.im * t1.im
        vmulps      ymm4, ymm4, ymm14           ; x8-x5.re * t1.re
        vmulps      ymm10, ymm10, ymm15         ; x8-x5.im * t1.im
        vmulps      ymm2, ymm2, oword ptr [rbx] ; x5-x2.re * t2.re
        vmulps      ymm8, ymm8, oword ptr [rbx+20h] ; x5-x2.im * t2.im
        vmulps      ymm5, ymm5, oword ptr [rbx] ; x9-x6.re * t2.re
        vmulps      ymm11, ymm11,oword ptr[rbx+20h] ; x9-x6.im * t2.im
        vaddps      ymm0, ymm0, ymm6
        vaddps      ymm3, ymm3, ymm9
        vaddps      ymm1, ymm1, ymm7
        vaddps      ymm4, ymm4, ymm10
        vaddps      ymm2, ymm2, ymm8
        vaddps      ymm5, ymm5, ymm11
        vaddps      ymm0, ymm0, ymm1
        vaddps      ymm3, ymm3, ymm4
        vaddps      ymm0, ymm0, ymm2
        vaddps      ymm3, ymm3, ymm5
        vmovaps     oword ptr [r10], ymm0       ; Y3-Y0
        vmovaps     oword ptr [r10+20h], ymm3   ; Y7-Y4
        add         r10, 40h
        cmp         r11, r13
        jl          doA03by8                    ; 43 instructions/8 points = ~5.5 ipp
try3tail:
        mov         r13,len
        and         r13,7
        je          exit3
do3tail:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        add         r11,8
        vpshufd     xmm3,xmm0,0B1h
        vpshufd     xmm4,xmm1,0B1h
        vpshufd     xmm5,xmm2,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+80h]
        vmulps      xmm1,xmm1,oword ptr [rbx+40h]
        vmulps      xmm2,xmm2,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx+0A0h]
        vmulps      xmm4,xmm4,oword ptr [rbx+60h]
        vmulps      xmm5,xmm5,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm2,xmm2,xmm4
        vaddps      xmm0,xmm0,xmm2
        vmovsd      qword ptr [r10],xmm0
```

```
        add         r10,8
        sub         r13,1
        ja          do3tail
exit3:
        REST_XMM
        REST_GPR
        ret
doU03by4:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovsd      xmm3,qword ptr [r11+18h]
        vmovups     xmm7,oword ptr [r11+20h]
        add         r11,20h
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vmovaps     xmm3,xmm2
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+80h]
        vmulps      xmm1,xmm1,oword ptr [rbx+40h]
        vmulps      xmm2,xmm2,oword ptr [rbx]
        vmulps      xmm4,xmm4,oword ptr [rbx+0A0h]
        vmulps      xmm5,xmm5,oword ptr [rbx+60h]
        vmulps      xmm6,xmm6,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm4
        vaddps      xmm5,xmm5,xmm6
        vaddps      xmm2,xmm2,xmm5
        vaddps      xmm0,xmm0,xmm2
        vmovaps     oword ptr [r10],xmm0
        vmovhlps    xmm4,xmm4,xmm3
        vmovlhps    xmm4,xmm4,xmm7
        vpshufd     xmm0,xmm3,0B1h
        vpshufd     xmm1,xmm4,0B1h
        vpshufd     xmm2,xmm7,0B1h
        vmulps      xmm3,xmm3,oword ptr [rbx+80h]
        vmulps      xmm4,xmm4,oword ptr [rbx+40h]
        vmulps      xmm7,xmm7,oword ptr [rbx]
        vmulps      xmm0,xmm0,oword ptr [rbx+0A0h]
        vmulps      xmm1,xmm1,oword ptr [rbx+60h]
        vmulps      xmm2,xmm2,oword ptr [rbx+20h]
        vaddps      xmm3,xmm3,xmm4
        vaddps      xmm7,xmm7,xmm0
        vaddps      xmm1,xmm1,xmm2
        vaddps      xmm7,xmm7,xmm1
        vaddps      xmm3,xmm3,xmm7
        vmovaps     oword ptr [r10+10h],xmm3
        add         r10,20h
        cmp         r11,r13
        jl          doU03by4
        jmp         try3tail

Order3:
        and         r13,-4
        je          try4tail
        lea         r13,[r11+8*r13]
        test        r11,1Fh
        jne         doU04by4
        vmovaps     ymm8, oword ptr [rbx+0C0h]  ; t0.re, t0.re, t0.re, t0.re, t0.re, t0.re,
t0.re, t0.re
        vmovaps     ymm9, oword ptr [rbx+0E0h]  ; t0.im,-t0.im, t0.im,-t0.im, t0.im,-t0.im,
t0.im,-t0.im
        vmovaps     ymm10, oword ptr [rbx+80h]  ; t1.re, t1.re, t1.re, t1.re, t1.re, t1.re,
t1.re, t1.re
        vmovaps     ymm11, oword ptr [rbx+0A0h] ; t1.im,-t1.im, t1.im,-t1.im, t1.im,-t1.im,
t1.im,-t1.im
        vmovaps     ymm12, oword ptr [rbx+40h]  ; t2.re, t2.re, t2.re, t2.re, t2.re, t2.re,
t2.re, t2.re
```

```
        vmovaps     ymm13, oword ptr [rbx+60h]  ; t2.im,-t2.im, t2.im,-t2.im, t2.im,-t2.im,
t2.im,-t2.im
        vmovaps     ymm14, oword ptr [rbx]      ; t3.re, t3.re, t3.re, t3.re, t3.re, t3.re,
t3.re, t3.re
        vmovaps     ymm15, oword ptr [rbx+20h]  ; t3.im,-t3.im, t3.im,-t3.im, t3.im,-t3.im,
t3.im,-t3.im
doA04by4:
        vmovaps     ymm0, oword ptr [r11]       ; x3.im, x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vmovaps     ymm4, oword ptr [r11+20h]   ; x7.im, x7.re, x6.im, x6.re, x5.im, x5.re,
x4.im, x4.re
        add         r11, 20h
        vperm2f128  ymm2, ymm0, ymm4, 21h       ; x5.im, x5.re, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm0, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vblendps    ymm3, ymm4, ymm2, 0CCh      ; x5.im, x5.re, x6.im, x6.re, x3.im, x3.re,
x4.im, x4.re
        vshufps     ymm3, ymm3, ymm3, 4Eh       ; x6.im, x6.re, x5.im, x5.re, x4.im, x4.re,
x3.im, x3.re
        vshufps     ymm4, ymm0, ymm0, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vshufps     ymm5, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vshufps     ymm6, ymm2, ymm2, 0B1h      ; x5.re, x5.im, x4.re, x4.im, x3.re, x3.im,
x2.re, x2.im
        vshufps     ymm7, ymm3, ymm3, 0B1h      ; x6.re, x6.im, x5.re, x5.im, x4.re, x4.im,
x3.re, x3.im
        vmulps      ymm0, ymm0, ymm8            ; x3-x0 * t0.re
        vmulps      ymm4, ymm4, ymm9            ; x3-x0 * t0.im
        vaddps      ymm0, ymm0, ymm4            ; x3-x0 * t0
        vmulps      ymm1, ymm1, ymm10           ; x4-x1 * t1.re
        vmulps      ymm5, ymm5, ymm11           ; x4-x1 * t1.im
        vaddps      ymm1, ymm1, ymm5            ; x4-x1 * t1
        vmulps      ymm2, ymm2, ymm12           ; x5-x2 * t2.re
        vmulps      ymm6, ymm6, ymm13           ; x5-x2 * t2.im
        vaddps      ymm2, ymm2, ymm6            ; x5-x2 * t2
        vmulps      ymm3, ymm3, ymm14           ; x6-x3 * t3.re
        vmulps      ymm7, ymm7, ymm15           ; x6-x3 * t3.im
        vaddps      ymm3, ymm3, ymm7            ; x6-x3 * t3
        vaddps      ymm0, ymm0, ymm1            ;
        vaddps      ymm2, ymm2, ymm3
        vaddps      ymm0, ymm0, ymm2
        vmovaps     oword ptr [r10], ymm0
        add         r10, 20h
        cmp         r11, r13
        jl          doA04by4                    ; 31 instructions per 4 points =~8 ipp
try4tail:
        mov         r13,len
        and         r13,3
        je          exit4
do4tail:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovsd      xmm3,qword ptr [r11+18h]
        add         r11,8
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+0C0h]
        vmulps      xmm4,xmm4,oword ptr [rbx+0E0h]
        vmulps      xmm1,xmm1,oword ptr [rbx+80h]
        vmulps      xmm5,xmm5,oword ptr [rbx+0A0h]
        vmulps      xmm2,xmm2,oword ptr [rbx+40h]
        vmulps      xmm6,xmm6,oword ptr [rbx+60h]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
```

```
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm6,xmm6,xmm7
        vaddps      xmm0,xmm0,xmm2
        vaddps      xmm4,xmm4,xmm6
        vaddps      xmm0,xmm0,xmm4
        vmovsd      qword ptr [r10],xmm0
        add         r10,8
        sub         r13,1
        ja          do4tail
exit4:
        REST_XMM
        REST_GPR
        ret
doU04by4:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovups     xmm3,oword ptr [r11+18h]
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+0C0h]
        vmulps      xmm4,xmm4,oword ptr [rbx+0E0h]
        vmulps      xmm1,xmm1,oword ptr [rbx+80h]
        vmulps      xmm5,xmm5,oword ptr [rbx+0A0h]
        vmulps      xmm2,xmm2,oword ptr [rbx+40h]
        vmulps      xmm6,xmm6,oword ptr [rbx+60h]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm6,xmm6,xmm7
        vaddps      xmm0,xmm0,xmm2
        vaddps      xmm4,xmm4,xmm6
        vaddps      xmm0,xmm0,xmm4
        vmovaps     oword ptr [r10],xmm0
        vmovsd      xmm0,qword ptr [r11+10h]
        vmovsd      xmm1,qword ptr [r11+18h]
        vmovsd      xmm2,qword ptr [r11+20h]
        vmovups     xmm3,oword ptr [r11+28h]
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        add         r11,20h
        vmovhlps    xmm1,xmm1,xmm0
        vmovlhps    xmm1,xmm1,xmm2
        vmovhlps    xmm3,xmm3,xmm2
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vpshufd     xmm7,xmm3,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx+0C0h]
        vmulps      xmm4,xmm4,oword ptr [rbx+0E0h]
        vmulps      xmm1,xmm1,oword ptr [rbx+80h]
        vmulps      xmm5,xmm5,oword ptr [rbx+0A0h]
        vmulps      xmm2,xmm2,oword ptr [rbx+40h]
        vmulps      xmm6,xmm6,oword ptr [rbx+60h]
        vmulps      xmm3,xmm3,oword ptr [rbx]
        vmulps      xmm7,xmm7,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm6,xmm6,xmm7
        vaddps      xmm0,xmm0,xmm2
```

```
        vaddps      xmm4,xmm4,xmm6
        vaddps      xmm0,xmm0,xmm4
        vmovaps     oword ptr [r10+10h],xmm0
        add         r10,20h
        cmp         r11,r13
        jl          doU04by4
        jmp         try4tail
OrderAny:
        mov         rax,order
        add         rax,rax
        add         rax,rax
        add         rax,rax
        lea         rbx,[rbx+8*rax]
        mov         pTaps,rbx
        add         order,1
        mov         r12,r11
        and         r13,-4
        je          tryAtail
        lea         r13,[r11+8*r13]
        test        r11,1Fh
        jne         nxtU2points
next2points:
        mov         r11,r12
        mov         rbx,pTaps
        mov         rax,order
        vxorps      ymm8, ymm8, ymm8
        add         r12,20h
doA0Aby4:
        vmovaps     ymm0, oword ptr [r11]       ; x3.im. x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vmovaps     ymm4, oword ptr [r11+20h]   ; x7.im. x7.re, x6.im, x6.re, x5.im, x5.re,
x4.im, x4.re
        add         r11, 20h
        sub         rax, 4
        vperm2f128  ymm2, ymm0, ymm4, 21h       ; x5.im, x5.re, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm0, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vblendps    ymm3, ymm4, ymm2, 0CCh      ; x5.im, x5.re, x6.im, x6.re, x3.im, x3.re,
x4.im, x4.re
        vshufps     ymm3, ymm3, ymm3, 4Eh       ; x6.im, x6.re, x5.im, x5.re, x4.im, x4.re,
x3.im, x3.re
        vshufps     ymm4, ymm0, ymm0, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vshufps     ymm5, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vshufps     ymm6, ymm2, ymm2, 0B1h      ; x5.re, x5.im, x4.re, x4.im, x3.re, x3.im,
x2.re, x2.im
        vshufps     ymm7, ymm3, ymm3, 0B1h      ; x6.re, x6.im, x5.re, x5.im, x4.re, x4.im,
x3.re, x3.im
        vmulps      ymm0, ymm0, oword ptr [rbx]     ; x3-x0 * t0.re
        vmulps      ymm4, ymm4, oword ptr [rbx+20h] ; x3-x0 * t0.im
        vaddps      ymm0, ymm0, ymm4                ; x3-x0 * t0
        vmulps      ymm1, ymm1, oword ptr [rbx-40h] ; x4-x1 * t1.re
        vmulps      ymm5, ymm5, oword ptr [rbx-20h] ; x4-x1 * t1.im
        vaddps      ymm1, ymm1, ymm5                ; x4-x1 * t1
        vmulps      ymm2, ymm2, oword ptr [rbx-80h] ; x5-x2 * t2.re
        vmulps      ymm6, ymm6, oword ptr [rbx-60h] ; x5-x2 * t2.im
        vaddps      ymm2, ymm2, ymm6                ; x5-x2 * t2
        vmulps      ymm3, ymm3, oword ptr [rbx-0C0h]; x6-x3 * t3.re
        vmulps      ymm7, ymm7, oword ptr [rbx-0A0h]; x6-x3 * t3.im
        sub         rbx, 100h
        vaddps      ymm3, ymm3, ymm7                ; x6-x3 * t3
        vaddps      ymm0, ymm0, ymm1                ;
        vaddps      ymm2, ymm2, ymm3
        vaddps      ymm0, ymm0, ymm2
        vaddps      ymm8, ymm0, ymm8
        cmp         rax, 4
        jae         doA0Aby4
```

```
        cmp         rax,3
        je          tail3tpA
        cmp         rax,2
        je          tail2tpA
        cmp         rax,1
        je          tail1tpA
wrResult:
        vmovaps     oword ptr [r10],ymm8
        add         r10,20h
        cmp         r12,r13
        jl          next2points
tryAtail:
        mov         r9, r12
        and         len, 3
nxtAtail:
        cmp         len,0
        jle         exitA
        mov         r12, r9
        mov         rax,order
        mov         rbx,pTaps
        vxorps      ymm7, ymm7, ymm7
nxtTap:
        vmovsd      xmm0,qword ptr [r12]
        add         r12,8
        vpshufd     xmm1,xmm0,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm1,xmm1,oword ptr [rbx+20h]
        sub         rbx,40h
        vaddps      xmm7,xmm7,xmm0
        vaddps      xmm7,xmm7,xmm1
        sub         rax,1
        ja          nxtTap
        vmovsd      qword ptr [r10],xmm7
        add         r10, 8
        add         r9, 8
        sub         len, 1
        jmp         nxtAtail
exitA:
        REST_XMM
        REST_GPR
        ret
tail3tpA:
        vmovaps     ymm3, oword ptr [r11]       ; x3.im. x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vmovaps     xmm4, oword ptr [r11+20h]   ;                               x5.im, x5.re,
x4.im, x4.re
        vperm2f128  ymm2, ymm3, ymm4, 21h       ; x5.im, x5.re, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm3, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vshufps     ymm4, ymm3, ymm3, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vshufps     ymm5, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vshufps     ymm6, ymm2, ymm2, 0B1h      ; x5.re, x5.im, x4.re, x4.im, x3.re, x3.im,
x2.re, x2.im
        vmulps      ymm3, ymm3, oword ptr [rbx]     ; x3-x0 * t0.re
        vmulps      ymm4, ymm4, oword ptr [rbx+20h] ; x3-x0 * t0.im
        vaddps      ymm3, ymm3, ymm4                ; x3-x0 * t0
        vmulps      ymm1, ymm1, oword ptr [rbx-40h] ; x4-x1 * t1.re
        vmulps      ymm5, ymm5, oword ptr [rbx-20h] ; x4-x1 * t1.im
        vaddps      ymm1, ymm1, ymm5                ; x4-x1 * t1
        vmulps      ymm2, ymm2, oword ptr [rbx-80h] ; x5-x2 * t2.re
        vmulps      ymm6, ymm6, oword ptr [rbx-60h] ; x5-x2 * t2.im
        vaddps      ymm2, ymm2, ymm6                ; x5-x2 * t2
        vaddps      ymm3, ymm3, ymm1                ;
        vaddps      ymm3, ymm3, ymm2
        vaddps      ymm8, ymm3, ymm8
        vmovaps     oword ptr [r10],ymm8
```

```
        add         r10,20h
        cmp         r12,r13
        jl          next2points
        jmp         tryAtail
tail2tpA:
        vmovaps     ymm3, oword ptr [r11]       ; x3.im. x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re

                    movlps     xmm4, qword ptr [r11+20h]  ;                          x4.im,
            x4.re
        vperm2f128  ymm2, ymm3, ymm4, 21h       ; xxxxx, xxxxx, x4.im, x4.re, x3.im, x3.re,
x2.im, x2.re
        vblendps    ymm1, ymm2, ymm3, 0CCh      ; x3.im, x3.re, x4.im, x4.re, x1.im, x1.re,
x2.im, x2.re
        vshufps     ymm1, ymm1, ymm1, 4Eh       ; x4.im, x4.re, x3.im, x3.re, x2.im, x2.re,
x1.im, x1.re
        vshufps     ymm4, ymm3, ymm3, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vshufps     ymm5, ymm1, ymm1, 0B1h      ; x4.re, x4.im, x3.re, x3.im, x2.re, x2.im,
x1.re, x1.im
        vmulps      ymm3, ymm3, oword ptr [rbx]     ; x3-x0 * t0.re
        vmulps      ymm4, ymm4, oword ptr [rbx+20h] ; x3-x0 * t0.im
        vaddps      ymm3, ymm3, ymm4                ; x3-x0 * t0
        vmulps      ymm1, ymm1, oword ptr [rbx-40h] ; x4-x1 * t1.re
        vmulps      ymm5, ymm5, oword ptr [rbx-20h] ; x4-x1 * t1.im
        vaddps      ymm1, ymm1, ymm5                ; x4-x1 * t1
        vaddps      ymm3, ymm3, ymm1                ;
        vaddps      ymm8, ymm3, ymm8
        vmovaps     oword ptr [r10],ymm8
        add         r10,20h
        cmp         r12,r13
        jl          next2points
        jmp         tryAtail
tail1tpA:
        vmovaps     ymm3, oword ptr [r11]       ; x3.im. x3.re, x2.im, x2.re, x1.im, x1.re,
x0.im, x0.re
        vshufps     ymm4, ymm3, ymm3, 0B1h      ; x3.re, x3.im, x2.re, x2.im, x1.re, x1.im,
x0.re, x0.im
        vmulps      ymm3, ymm3, oword ptr [rbx] ; x3-x0 * t0.re
        vmulps      ymm4, ymm4, oword ptr [rbx+20h] ; x3-x0 * t0.im
        vaddps      ymm3, ymm3, ymm4                ; x3-x0 * t0
        vaddps      ymm8, ymm3, ymm8
        vmovaps     oword ptr [r10],ymm8
        add         r10,20h
        cmp         r12,r13
        jl          next2points
        jmp         tryAtail

nxtU2points:
        mov         r11, r12
        mov         rbx, pTaps
        mov         rax, order
        vxorps      ymm7, ymm7, ymm7
        add         r12, 10h
doU0Aby4:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovsd      xmm3,qword ptr [r11+18h]
        movhps      xmm3,qword ptr [r11+20h]
        add         r11,20h
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vpshufd     xmm4,xmm0,0B1h
        vpshufd     xmm5,xmm1,0B1h
        vpshufd     xmm6,xmm2,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm4,xmm4,oword ptr [rbx+20h]
        vmulps      xmm1,xmm1,oword ptr [rbx-40h]
```

```
        vmulps      xmm5,xmm5,oword ptr [rbx-20h]
        vmulps      xmm2,xmm2,oword ptr [rbx-80h]
        vmulps      xmm6,xmm6,oword ptr [rbx-60h]
        vaddps      xmm0,xmm0,xmm1
        vpshufd     xmm1,xmm3,0B1h
        vmulps      xmm3,xmm3,oword ptr [rbx-0C0h]
        sub         rax,4
        vmulps      xmm1,xmm1,oword ptr [rbx-0A0h]
        sub         rbx,100h
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm6,xmm6,xmm1
        vaddps      xmm0,xmm0,xmm2
        vaddps      xmm4,xmm4,xmm6
        vaddps      xmm7,xmm7,xmm0
        vaddps      xmm7,xmm7,xmm4
        cmp         rax,4
        jae         doU0Aby4
        cmp         rax,3
        je          tail3tpU
        cmp         rax,2
        je          tail2tpU
        cmp         rax,1
        je          tail1tpU
        vmovaps     oword ptr [r10],xmm7
        add         r10,10h
        cmp         r12,r13
        jl          nxtU2points
        jmp         tryAtail
tail3tpU:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovsd      xmm2,qword ptr [r11+10h]
        vmovsd      xmm3,qword ptr [r11+18h]
        vmovlhps    xmm0,xmm0,xmm1
        vmovlhps    xmm1,xmm1,xmm2
        vmovlhps    xmm2,xmm2,xmm3
        vpshufd     xmm3,xmm0,0B1h
        vpshufd     xmm4,xmm1,0B1h
        vpshufd     xmm5,xmm2,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx+20h]
        vmulps      xmm1,xmm1,oword ptr [rbx-40h]
        vmulps      xmm4,xmm4,oword ptr [rbx-20h]
        vmulps      xmm2,xmm2,oword ptr [rbx-80h]
        vmulps      xmm5,xmm5,oword ptr [rbx-60h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm2,xmm2,xmm3
        vaddps      xmm4,xmm4,xmm5
        vaddps      xmm0,xmm0,xmm2
        vaddps      xmm7,xmm7,xmm4
        vaddps      xmm7,xmm7,xmm0
        vmovaps     oword ptr [r10],xmm7
        add         r10,10h
        cmp         r12,r13
        jl          nxtU2points
        jmp         tryAtail
tail2tpU:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovhps     xmm1,xmm1,qword ptr [r11+10h]
        vmovlhps    xmm0,xmm0,xmm1
        vpshufd     xmm3,xmm0,0B1h
        vpshufd     xmm4,xmm1,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx+20h]
        vmulps      xmm1,xmm1,oword ptr [rbx-40h]
        vmulps      xmm4,xmm4,oword ptr [rbx-20h]
        vaddps      xmm0,xmm0,xmm1
        vaddps      xmm3,xmm3,xmm4
        vaddps      xmm0,xmm0,xmm3
```

```
        vaddps      xmm7,xmm7,xmm0
        vmovaps     oword ptr [r10],xmm7
        add         r10,10h
        cmp         r12,r13
        jl          nxtU2points
        jmp         tryAtail
tail1tpU:
        vmovsd      xmm0,qword ptr [r11]
        vmovsd      xmm1,qword ptr [r11+8]
        vmovlhps    xmm0,xmm0,xmm1
        vpshufd     xmm3,xmm0,0B1h
        vmulps      xmm0,xmm0,oword ptr [rbx]
        vmulps      xmm3,xmm3,oword ptr [rbx+20h]
        vaddps      xmm0,xmm0,xmm3
        vaddps      xmm7,xmm7,xmm0
        vmovaps     oword ptr [r10],xmm7
        add         r10,10h
        cmp         r12,r13
        jl          nxtU2points
        jmp         tryAtail
ownsIIRxAR_32fc ENDP
```

Assembler Function for A Coefficients

```
;###############################################################################
;##   ownsIIRyAR_32fc( Ipp32fc *pSrc, Ipp32fc *pDst, int len, Ipp32f *pTaps  ##
;##                                  int order );                             ##
;###############################################################################
;   Declare input variables

pSrc        equ rdi
pDst        equ rsi
len         equ rdx
pTaps       equ rcx
order       equ r8

ALIGN 16

; Lib = E9
; Caller = ippsIIR_32fc
; Caller = ippsIIR_32fc_I

ownsIIRyAR_32fc PROC PUBLIC FRAME
        USES_GPR rbx, rsi, rdi,r12,r13
        LOCAL_FRAME = 0
        USES_XMM xmm6, xmm7, xmm8, xmm9, xmm10, xmm11, xmm12, xmm13, xmm14, xmm15
        COMP_ABI 5
        movsxd      r8,r8d
        mov         r12,pTaps
        mov         r10,pSrc
        mov         r11,pDst
        mov         r13,len
        mov         rax,order
        cmp         rax,1
        je          Order1
        cmp         rax,2
        je          Order2
        jmp         OrderAny
Order1:
        vbroadcastsd  ymm0, [r11]               ;y0.im,y0.re,y0.im,y0.re,y0.im,y0.re,y0.im,y0.re
        add         r11, 8
        and         r13, -4
        je          try1by1
        lea         r13, [r11+8*r13]
        vmovaps     ymm8, oword ptr [r12]      ;A1[3],A1[2],A1[1],A1[0] coefficients for "re" mul
        vmovaps     ymm9, oword ptr [r12+20h]  ;A1[3],A1[2],A1[1],A1[0] coefficients for "im" mul
        vmovaps     ymm10, oword ptr [r12+40h] ;A1[2],A1[1],A1[0],00000 coefficients for "re" mul
        vmovaps     ymm11, oword ptr [r12+60h] ;A1[2],A1[1],A1[0],00000 coefficients for "im" mul
```

```
        vmovaps     ymm12, oword ptr [r12+80h] ;A1[1],A1[0],00000,00000 coefficients for "re" mul
        vmovaps     ymm13, oword ptr [r12+0A0h];A1[1],A1[0],00000,00000 coefficients for "im" mul
        vmovaps     ymm14, oword ptr [r12+0C0h];A1[0],00000,00000,00000 coefficients for "re" mul
        vmovaps     ymm15, oword ptr [r12+0E0h];A1[0],00000,00000,00000 coefficients for "im" mul
do1by4:
        vperm2f128  ymm0, ymm0, ymm0, 11h      ;y0.im,y0.re,xxxxx,xxxxx,y0.im,y0.re,xxxxx,xxxxx
        vshufps     ymm2, ymm0, ymm0, 0FFh     ;y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im
        vshufps     ymm0, ymm0, ymm0, 0AAh     ;y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re
        vmulps      ymm0, ymm0, ymm8           ; y.re * A2A0
        vmulps      ymm2, ymm2, ymm9           ; y.im * A3A1
        vaddps      ymm4, ymm0, ymm2           ; y.re * A2A0 + y.im * A3A1
        vmovups     ymm7, oword ptr [r10]      ;x3.im,x3.re,x2.im,x2.re,x1.im,x1.re,x0.im,x0.re
        vbroadcastss ymm0, [r10]               ;x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re
        vbroadcastss ymm1, [r10+4]             ;x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im
        vmulps      ymm0, ymm0, ymm10          ; x0.re * pA[16*order+0]
        vmulps      ymm1, ymm1, ymm11          ; x0.im * pA[16*order+8]
        vbroadcastss ymm2, [r10+8]             ;x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re
        vbroadcastss ymm3, [r10+12]            ;x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im
        vaddps      ymm0, ymm0, ymm1           ; x0.re * pA + x0.im * pA
        vmulps      ymm2, ymm2, ymm12          ; x1.re * pA[16*order+16]
        vmulps      ymm3, ymm3, ymm13          ; x1.im * pA[16*order+24]
        vaddps      ymm7, ymm4, ymm7           ; add all 4 x values (last column of coefficients)
        vbroadcastss ymm4, [r10+16]            ;x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re
        vbroadcastss ymm5, [r10+20]            ;x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im
        vaddps      ymm2, ymm2, ymm3
        vmulps      ymm4, ymm4, ymm14          ; x2.re * pA[16*order+32]
        vmulps      ymm5, ymm5, ymm15          ; x2.im * pA[16*order+40]
        add         r10, 20h
        vaddps      ymm4, ymm4, ymm5
        vaddps      ymm0, ymm2, ymm0           ; accumulator
        vaddps      ymm0, ymm0, ymm4
        vaddps      ymm0, ymm7, ymm0
        vmovups     oword ptr [r11], ymm0
        add         r11, 20h
        cmp         r11, r13
        jl          do1by4
try1by1:
        mov         r13, len
        and         r13, 3
        je          exit1
        vextractf128 xmm2, ymm0, 1
        vmovhlps    xmm2, xmm2, xmm2           ; xxxxx, xxxxx, y0.im, y0.re
do1by1:
        vmovsd      xmm5, qword ptr [r10]      ; x.im,x.re
        add         r10, 8
        vpshufd     xmm4, xmm2, 55h            ; im
        vpshufd     xmm2, xmm2, 0              ; re
        vmulps      xmm2, xmm2, oword ptr [r12]
        vmulps      xmm4, xmm4, oword ptr [r12+20h]
        vaddps      xmm5, xmm5, xmm4
        vaddps      xmm2, xmm2, xmm5
        vmovsd      qword ptr [r11], xmm2
        add         r11, 8
        sub         r13, 1
        ja          do1by1
exit1:
        REST_XMM
        REST_GPR
        ret
Order2:
        vmovups        xmm0, [r11]             ;xxxxx,xxxxx,xxxxx,xxxxx,y1.im,y1.re,y0.im,y0.re
        vperm2f128  ymm0, ymm0, ymm0, 0        ;y1.im,y1.re,y0.im,y0.re,y1.im,y1.re,y0.im,y0.re
        add         r11, 10h
        and         r13, -4
        je          try2by1
        lea         r13, [r11+8*r13]
        vmovaps     ymm10, oword ptr [r12+80h] ;A1[2],A1[1],A1[0],00000 coefficients for "re" mul
        vmovaps     ymm11, oword ptr [r12+0A0h];A1[2],A1[1],A1[0],00000 coefficients for "im" mul
        vmovaps     ymm12, oword ptr [r12+0C0h];A1[1],A1[0],00000,00000 coefficients for "re" mul
        vmovaps     ymm13, oword ptr [r12+0E0h];A1[1],A1[0],00000,00000 coefficients for "im" mul
        vmovaps     ymm14, oword ptr [r12+100h];A1[0],00000,00000,00000 coefficients for "re" mul
```

```
        vmovaps     ymm15, oword ptr [r12+120h];A1[0],00000,00000,00000 coefficients for "im" mul
do2by4:
        vperm2f128  ymm0, ymm0, ymm0, 11h      ;y1.im,y1.re,y0.im,y0.re,y1.im,y1.re,y0.im,y0.re
        vshufps     ymm3, ymm0, ymm0, 55h      ;y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im
        vshufps     ymm2, ymm0, ymm0, 0        ;y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re
        vshufps     ymm1, ymm0, ymm0, 0FFh     ;y1.im,y1.im,y1.im,y1.im,y1.im,y1.im,y1.im,y1.im
        vshufps     ymm0, ymm0, ymm0, 0AAh     ;y1.re,y1.re,y1.re,y1.re,y1.re,y1.re,y1.re,y1.re
        vmulps      ymm0, ymm0, oword ptr [r12]     ; A1[3],A1[2],A1[1],A1[0].re * y1.re
        vmulps      ymm1, ymm1, oword ptr [r12+20h] ; A1[3],A1[2],A1[1],A1[0].im * y1.im
        vmulps      ymm2, ymm2, oword ptr [r12+40h] ; A2[3],A2[2],A2[1],A2[0].re * y0.re
        vmulps      ymm3, ymm3, oword ptr [r12+60h] ; A2[3],A2[2],A2[1],A2[0].im * y0.im
        vaddps      ymm0, ymm0, ymm1
        vaddps      ymm2, ymm3, ymm2
        vaddps      ymm0, ymm2, ymm0           ; accumulator
        vmovups     ymm7, oword ptr [r10]      ;x3.im,x3.re,x2.im,x2.re,x1.im,x1.re,x0.im,x0.re
        add         r10, 20h
        vperm2f128  ymm6, ymm7, ymm7, 0        ;x1.im,x1.re,x0.im,x0.re,x1.im,x1.re,x0.im,x0.re
        vperm2f128  ymm5, ymm7, ymm7, 11h      ;x3.im,x3.re,x2.im,x2.re,x3.im,x3.re,x2.im,x2.re
        vshufps     ymm2, ymm6, ymm6, 55h      ;x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im
        vshufps     ymm1, ymm6, ymm6, 0        ;x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re
        vmulps      ymm1, ymm1, ymm10          ; x0.re * pA[16*order+0]
        vmulps      ymm2, ymm2, ymm11          ; x0.im * pA[16*order+8]
        vaddps      ymm1, ymm1, ymm2           ; x0.re * pA + x0.im * pA
        vaddps      ymm0, ymm0, ymm1           ; accumulator
        vaddps      ymm0, ymm0, ymm7           ; add all 4 x values (last column of coefficients)
        vshufps     ymm1, ymm6, ymm6, 0AAh     ;x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re
        vshufps     ymm2, ymm6, ymm6, 0FFh     ;x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im
        vmulps      ymm1, ymm1, ymm12          ; x1.re * pA[16*order+16]
        vmulps      ymm2, ymm2, ymm13          ; x1.im * pA[16*order+24]
        vshufps     ymm3, ymm5, ymm5, 0        ;x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re
        vshufps     ymm4, ymm5, ymm5, 55h      ;x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im
        vaddps      ymm1, ymm1, ymm2
        vmulps      ymm3, ymm3, ymm14          ; x2.re * pA[16*order+32]
        vmulps      ymm4, ymm4, ymm15          ; x2.im * pA[16*order+40]
        vaddps      ymm0, ymm1, ymm0
        vaddps      ymm3, ymm3, ymm4
        vaddps      ymm0, ymm0, ymm3
        vmovups     oword ptr [r11], ymm0
        add         r11, 20h
        cmp         r11, r13
        jl          do2by4
try2by1:
        mov         r13,len
        and         r13,3
        je          exit2
        vperm2f128  ymm0, ymm0, ymm0, 11h      ;y1.im,y1.re,y0.im,y0.re,y1.im,y1.re,y0.im,y0.re
        vmovhlps    xmm0, xmm0, xmm0           ;y1.im,y1.re,y1.im,y1.re
do2by1:
        vmovsd      xmm2, qword ptr [r11-10h]  ; y0.im, y0.re
        vmovsd      xmm5, qword ptr [r10]      ; x.im,x.re
        add         r10, 8
        vpshufd     xmm1, xmm0, 55h            ; y1.im
        vpshufd     xmm0, xmm0, 0              ; y1.re
        vpshufd     xmm3, xmm2, 55h            ; y0.im
        vpshufd     xmm2, xmm2, 0              ; y0.re
        vmulps      xmm0, xmm0, oword ptr [r12]
        vmulps      xmm1, xmm1, oword ptr [r12+20h]
        vmulps      xmm2, xmm2, oword ptr [r12+40h]
        vmulps      xmm3, xmm3, oword ptr [r12+60h]
        vaddps      xmm0, xmm0, xmm1
        vaddps      xmm2, xmm2, xmm3
        vaddps      xmm0, xmm0, xmm5
        vaddps      xmm0, xmm0, xmm2
        vmovsd      qword ptr [r11], xmm0
        add         r11, 8
        sub         r13, 1
        ja          do2by1
exit2:
        REST_XMM
        REST_GPR
        ret
```

```
OrderAny:
        mov         rax, order
        shl         rax, 3
        lea         rbx, [r12+8*rax]
        mov         r12, rbx
        mov         pTaps, r12
        and         r13, -4
        je          tryAby1
        lea         rax, [r11+8*r13]
        vmovaps     ymm10, oword ptr [rbx]     ;A1[2],A1[1],A1[0],00000 coefficients for "re" mul
        vmovaps     ymm11, oword ptr [rbx+20h] ;A1[2],A1[1],A1[0],00000 coefficients for "im" mul
        vmovaps     ymm12, oword ptr [rbx+40h] ;A1[1],A1[0],00000,00000 coefficients for "re" mul
        vmovaps     ymm13, oword ptr [rbx+60h] ;A1[1],A1[0],00000,00000 coefficients for "im" mul
        vmovaps     ymm14, oword ptr [rbx+80h] ;A1[0],00000,00000,00000 coefficients for "re" mul
        vmovaps     ymm15, oword ptr [rbx+0A0h];A1[0],00000,00000,00000 coefficients for "im" mul
nextApoints:
        mov         r12, pTaps
        mov         r13, order
        vxorps      ymm8, ymm8, ymm8
doAby2:
        sub         r12, 80h
        vbroadcastss ymm0, [r11]               ;y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re
        vbroadcastss ymm1, [r11+4]             ;y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im
        vbroadcastss ymm2, [r11+8]             ;y1.re,y1.re,y1.re,y1.re,y1.re,y1.re,y1.re,y1.re
        vbroadcastss ymm3, [r11+12]            ;y1.im,y1.im,y1.im,y1.im,y1.im,y1.im,y1.im,y1.im
        add         r11, 10h
        vmulps      ymm2, ymm2, oword ptr [r12]     ; A1[3],A1[2],A1[1],A1[0].re * y1.re
        vmulps      ymm3, ymm3, oword ptr [r12+20h] ; A1[3],A1[2],A1[1],A1[0].im * y1.im
        vmulps      ymm0, ymm0, oword ptr [r12+40h] ; A2[3],A2[2],A2[1],A2[0].re * y0.re
        vmulps      ymm1, ymm1, oword ptr [r12+60h] ; A2[3],A2[2],A2[1],A2[0].im * y0.im
        vaddps      ymm2, ymm3, ymm2
        vaddps      ymm0, ymm0, ymm1
        vaddps      ymm8, ymm2, ymm8               ; accumulator
        vaddps      ymm8, ymm0, ymm8
        sub         r13, 2
        cmp         r13, 1
        ja          doAby2
        je          doAby1
tailAx:
        vmovaps     ymm7, oword ptr [r10]      ;x3.im,x3.re,x2.im,x2.re,x1.im,x1.re,x0.im,x0.re
        vbroadcastss ymm0, [r10]               ;x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re
        vbroadcastss ymm1, [r10+4]             ;x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im
        vmulps      ymm0, ymm0, ymm10          ; x0.re * pA[16*order+0]
        vmulps      ymm1, ymm1, ymm11          ; x0.im * pA[16*order+8]
        vbroadcastss ymm2, [r10+8]             ;x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re
        vbroadcastss ymm3, [r10+12]            ;x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im
        vaddps      ymm0, ymm0, ymm1           ; x0.re * pA + x0.im * pA
        vmulps      ymm2, ymm2, ymm12          ; x1.re * pA[16*order+16]
        vmulps      ymm3, ymm3, ymm13          ; x1.im * pA[16*order+24]
        vaddps      ymm8, ymm8, ymm7           ; add all 4 x values (last column of coefficients)
        vbroadcastss ymm4, [r10+16]            ;x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re
        vbroadcastss ymm5, [r10+20]            ;x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im
        vaddps      ymm2, ymm2, ymm3
        vmulps      ymm4, ymm4, ymm14          ; x2.re * pA[16*order+32]
        vmulps      ymm5, ymm5, ymm15          ; x2.im * pA[16*order+40]
        add         r10, 20h
        vaddps      ymm4, ymm4, ymm5
        vaddps      ymm0, ymm2, ymm0           ; accumulator
        vaddps      ymm0, ymm0, ymm4
        vaddps      ymm8, ymm8, ymm0
        vmovups     oword ptr [r11],ymm8
        add         pDst, 20h
        mov         r11, pDst
        cmp         r11, rax
        jl          nextApoints
tryAby1:
        mov         rax, len
        and         rax, 3
        je          exitA
nxtApoint:
        mov         r13, order
```

```
        mov         r11, pDst
        mov         r12, pTaps
        vxorps      ymm7, ymm7, ymm7
nxtAtap:
        sub         r12, 40h
        vmovsd      xmm0, qword ptr [r11]
        add         r11, 8
        vpshufd     xmm1, xmm0, 55h            ; im,im
        vpshufd     xmm0, xmm0, 0              ; re,re
        vmulps      xmm0, xmm0, oword ptr [r12]
        vmulps      xmm1, xmm1, oword ptr [r12+20h]
        vaddps      xmm0, xmm0, xmm1
        vaddps      xmm7, xmm7, xmm0
        sub         r13, 1
        ja          nxtAtap
        vmovsd      xmm0, qword ptr [r10]
        add         r10, 8
        vaddps      xmm7, xmm7, xmm0
        vmovsd      qword ptr [r11], xmm7
        add         pDst, 8
        sub         rax, 1
        ja          nxtApoint
exitA:
        REST_XMM
        REST_GPR
        ret
doAby1:
        vmovaps     ymm7, oword ptr [r10]      ;x3.im,x3.re,x2.im,x2.re,x1.im,x1.re,x0.im,x0.re
        sub         r12, 40h
        vbroadcastss ymm0, [r11]               ;y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re,y0.re
        vbroadcastss ymm1, [r11+4]             ;y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im,y0.im
        add         r11, 8
        vmulps      ymm0, ymm0, oword ptr [r12]     ; A1[3],A1[2],A1[1],A1[0].re * y0.re
        vmulps      ymm1, ymm1, oword ptr [r12+20h] ; A1[3],A1[2],A1[1],A1[0].im * y0.im
        vaddps      ymm0, ymm0, ymm1
        vaddps      ymm8, ymm0, ymm8
        vbroadcastss ymm1, [r10]               ;x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re,x0.re
        vbroadcastss ymm2, [r10+4]             ;x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im,x0.im
        vmulps      ymm3, ymm1, ymm10          ; x0.re * pA[16*order+0]
        vmulps      ymm4, ymm2, ymm11          ; x0.im * pA[16*order+8]
        vbroadcastss ymm1, [r10+8]             ;x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re,x1.re
        vbroadcastss ymm2, [r10+12]            ;x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im,x1.im
        vaddps      ymm3, ymm4, ymm3           ; x0.re * pA + x0.im * pA
        vaddps      ymm8, ymm8, ymm3           ; accumulator
        vmulps      ymm3, ymm1, ymm12          ; x1.re * pA[16*order+16]
        vmulps      ymm4, ymm2, ymm13          ; x1.im * pA[16*order+24]
        vaddps      ymm8, ymm8, ymm7           ; add all 4 x values (last column of coefficients)
        vbroadcastss ymm1, [r10+16]            ;x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re,x2.re
        vbroadcastss ymm2, [r10+20]            ;x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im,x2.im
        vaddps      ymm3, ymm4, ymm3
        vmulps      ymm1, ymm1, ymm14          ; x2.re * pA[16*order+32]
        vmulps      ymm2, ymm2, ymm15          ; x2.im * pA[16*order+40]
        add         r10, 20h
        vaddps      ymm8, ymm3, ymm8
        vaddps      ymm1, ymm1, ymm2
        vaddps      ymm8, ymm8, ymm1
        vmovups     oword ptr [r11],ymm8
        add         pDst, 20h
        mov         r11, pDst
        cmp         r11, rax
        jl          nextApoints
        jmp         tryAby1
ownsIIRyAR_32fc ENDP
```

About the Author

Igor Astakhov is a Department Software Engineer Manager with the Software Solutions Group (Visual Computing Software Division, CIP, and Intel IPP). He is focused on Intel IPP libraries development for all existing and future Intel Architectures. Igor has been at Intel for five years. His email is igor.astakhov@intel.com.